Annex A: Project en software engineering vereisten

Inhoud

1.1 Inleiding

ASTRON is op zoek naar een partner voor het verrichten van front-end software development diensten. Onder deze diensten verstaan we het ontwikkelen en testen (middels unit tests) van software gericht op een webapplicatie voor het bedienen van radiotelescopen middels de inzet van 3 development rollen: front-end development lead, lead developer, en senior developer. De werkzaamheden zullen plaatsvinden in een multidisciplinair agile ontwikkelteam waarbinnen ook de backend van het systeem ontwikkeld wordt. Dit betekend dat de ontwikkelaars meedraaien in alle scrum-events van het team.

1.2 Inleiding tot LOFAR

LOFAR (Low Frequency Array) is 's werelds grootste en meest gevoelige laagfrequente radiotelescoop. Het is ontworpen, gebouwd en wordt nu beheerd door ASTRON, het Nederlands Instituut voor Radioastronomie.

LOFAR, dat 52 antennestations in acht Europese landen verbindt met krachtige computers in Groningen, Nederland, werkt op de laagste radiofrequenties die vanaf de aarde kunnen worden waargenomen. In tegenstelling tot klassieke schoteltelescopen is LOFAR een multifunctioneel sensornetwerk, met een innovatieve computer- en netwerkinfrastructuur die extreem grote datavolumes aankan. Het lange termijnarchief bevat momenteel meer dan 40 petabyte aan astronomische gegevens.

De kern van LOFAR bestaat uit een netwerk van vierentwintig stations geconcentreerd in een gebied van drie kilometer in het noordoosten van Nederland; er zijn nog eens veertien stations in andere gebieden in Nederland. Het netwerk is in de loop der jaren gegroeid, waardoor LOFAR een echt pan-Europese onderzoek infrastructuur is geworden met zes stations in Duitsland, drie in Polen en één in Frankrijk, Ierland, Zweden, het Verenigd Koninkrijk, en de volgende wordt geleverd aan Letland in zomer 2019. Discussies over nieuwe stations in bijvoorbeeld Italië en Bulgarije zijn aan de gang. De grootste afstand tussen stations wordt de 'baseline' genoemd en nadert nu de 2000 km. Deze basislijn bepaalt de resolutie, of gedetailleerd inzoomvermogen, van de LOFAR-telescoop. Het totale aantal stations bepaalt de gevoeligheid voor de zwakste radio-emissie uit de lucht. Dit alles gebeurt op het dichtbevolkte Europese continent: het is een bewijs van het brein achter de continu verbeterende signaalverwerkingsalgoritmen dat dit haalbaar is.

Toegang tot LOFAR-observatie staat open voor wetenschappers van over de hele wereld volgens een competitief peer-reviewed proces dat in de astronomiediscipline 'open skies' wordt genoemd.

1.3 Inleiding tot het Telescope Manager-specificatiesysteem

Het Telescope Manager-specificatiesysteem (TMSS) is de volgende generatie softwaresuite voor het specificeren, beheren en plannen van LOFAR-observatie-, verwerkings- en opnametaken.

De belangrijkste doelstelling van TMSS is:

- Ondersteuning bieden voor specificatie, administratie, gegevensbeheer en planning om LOFAR Survey Use Cases uit te voeren. Het implementeert met name ondersteuning voor de volgende use-cases:
    - DUPLLO-UC1: DUPLLO HBA+ LBA survey;
    - DUPLLO-UC3: Beamformed pulsar survey data;
    - Commensal DUPLLO-UC1, DUPLLO-UC3;

De huidige software voor het systeem moet worden vervangen om het hoofddoel van TMSS te bereiken; het herbouwen van de huidige software voor een nieuw, schaalbaar ontwerp en technologie garandeert aanpassing van de DUPLLO-gebruiksscenario's.

TMSS zal de volgende voordelen realiseren:

- Ondersteuning voor LOFAR2.0-gebruiksscenario's;
- Efficiënte LOFAR-operaties;[1]
- Verbeterde aanpasbaarheid en onderhoudbaarheid van software.

[1] Gestroomlijnde LOFAR-operaties waarbij minder middelen nodig zijn om de telescoop op het huidige niveau te laten werken.

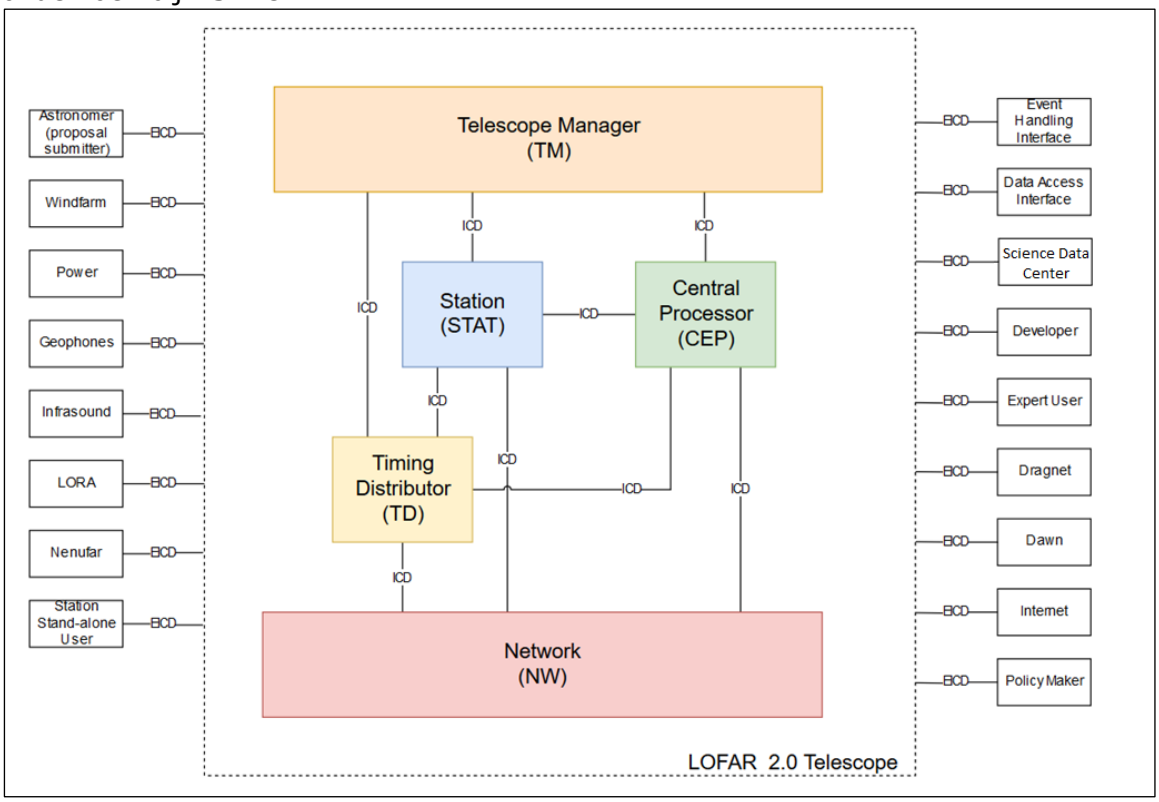
1.4 Projectbenadering

1.4.1 Product-centric

Visie en strategie voor een bedrijf kritisch product

Een product-centric benadering geeft het management van ASTRON een direct zicht op welke productkenmerken de waarde van TMSS maximaliseren, tegen welke kosten ze kunnen worden geproduceerd, en duidelijke verantwoordelijkheid van de producteigenaar. Product-centric benaderingen maken het gemakkelijker om snel te innoveren.[2]

TMSS is onderdeel van de Telescope Manager van LOFAR2.0 en kan daarom worden aangemerkt als mission critical. Het astronomische onderzoek is afhankelijk van de sterke punten van de LOFAR-faciliteit en de samenstellende delen zoals TMSS. Het uitvallen van TMSS zal ernstige gevolgen hebben voor de activiteiten van LOFAR en het astronomisch onderzoek bij ASTRON.



TMSS sluit volledig aan bij de visie en strategie van DUPLLO en LOFAR2.0. DUPPLO (de Digital Upgrade for Premier LOFAR Low band Observing) zal de gevoeligheid en het ontdekkingsvermogen van de Low Frequency Array (LOFAR) bij de laagste vanaf de aarde zichtbare radiofrequenties aanzienlijk verbeteren. LOFAR is 's werelds grootste en meest gevoelige laagfrequente radiotelescoop. LOFAR2.0 garandeert geavanceerde wetenschap voor het volgende decennium en verankert een ultramoderne, unieke, zeer productieve telescoop voor 2020-2030.

[2] "Making the shift from project to product is comparable to creating a product-centric technology organization, like becoming a software or digital business", een quote van Lars van Dam, Vice President Analyst, Gartner, February 2019

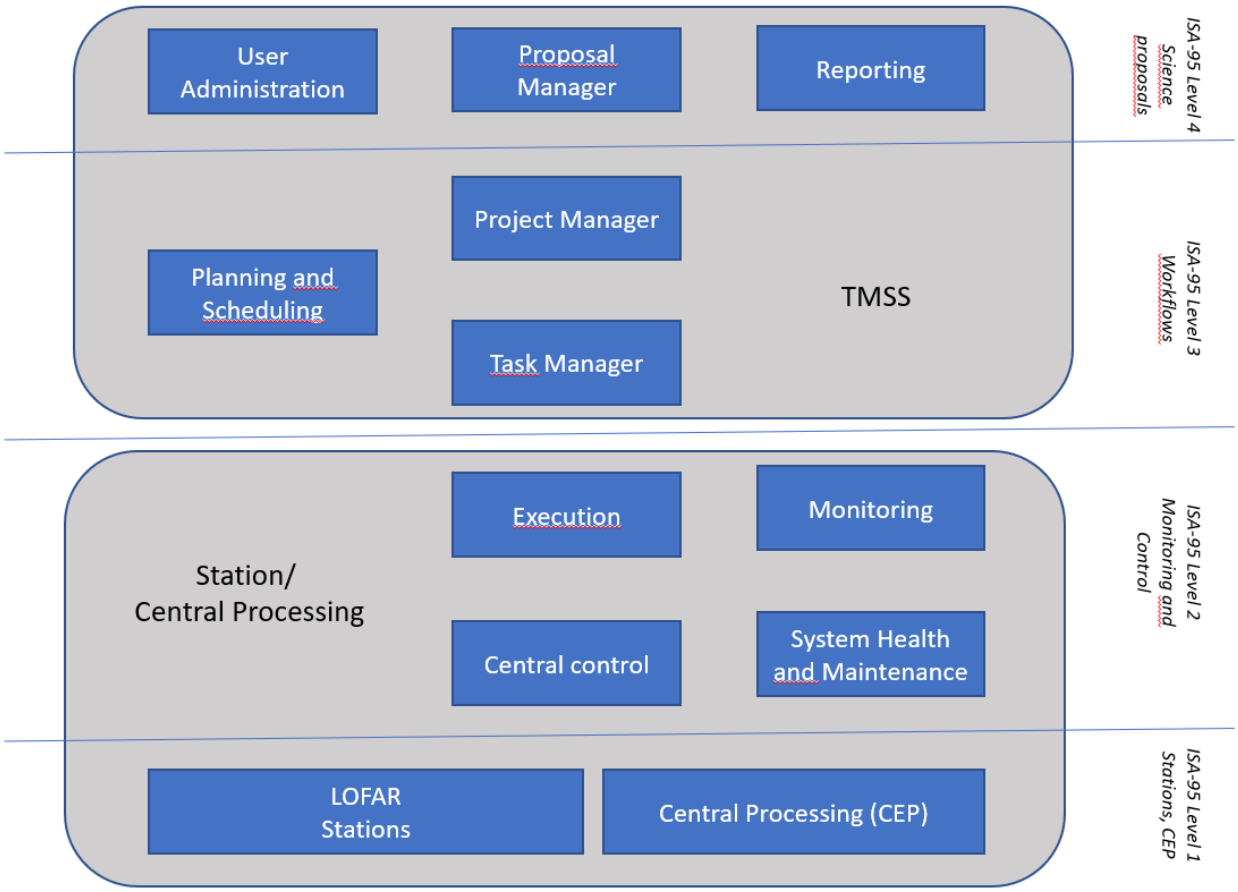
1.4.2 Systems Engineering

Vereisten / ontwerpbeperkingen / ontwerpbeslissingen

ASTRON gebruikt Systems Engineering (SE) voor het ontwerp van LOFAR2.0. Door gebruik te maken van SE kan LOFAR2.0 worden opgedeeld in kleine - en overzichtelijke - delen.[3]

De architecten van LOFAR2.0 gebruiken ISA-95 om de LOFAR2.0-architectuur te structureren. ISA-95 is de internationale standaard voor de integratie van bedrijfs- en controlesystemen. ISA-95 bestaat uit modellen en terminologie. Door gebruik te maken van deze standaard kan ASTRON een perspectief bieden op systeemengineering en systeemintegratie. Met deze modellen kan worden bepaald welke informatie tussen subsystemen en componenten moet worden uitgewisseld. De ISA-95-standaard helpt bij het definiëren van grenzen tussen systemen.

De softwarecomponenten van TMSS bevinden zich op de bovenste lagen van de ISA 95-standaard. De softwarecomponenten van TMSS op ISA-95 Level 3 werken samen met de TM-softwarecomponenten op IS-95 Level 2.



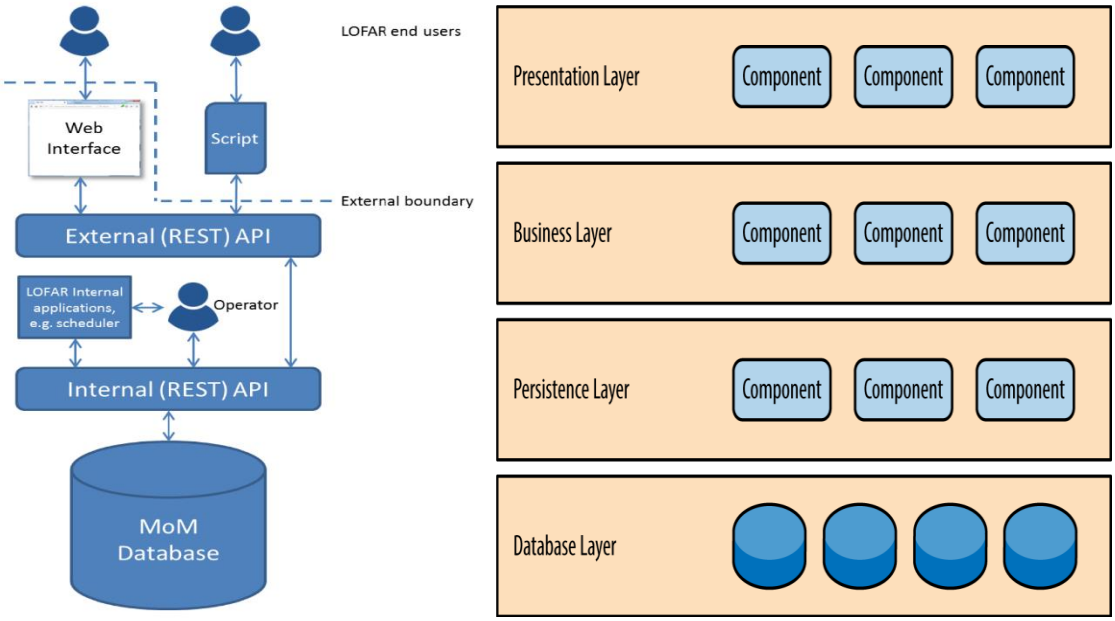
De systems engineer bij LOFAR2.0 en de software architect bij TMSS moeten nauw samenwerken. Dit is terug te vinden in het organogram van dit project (zie het volgende hoofdstuk).

[3] Tegenwoordig is de methode vastgelegd in een officiële norm, namelijk de ISO 15288.

1.4.3 Software engineering

Eén database, één applicatie, één GUI, één taal

Layered software architecture
De softwarecomponenten van TMSS worden gecombineerd in een applicatie met een gelaagde softwarearchitectuur. Dit patroon is de de facto standaard voor de meeste toepassingen en is daarom algemeen bekend bij de meeste architecten, ontwerpers en ontwikkelaars. Het gelaagde architectuurpatroon sluit nauw aan bij de traditionele IT-communicatie- en organisatiestructuren die in de meeste bedrijven worden aangetroffen, waardoor het een natuurlijke keuze is voor de meeste inspanningen voor de ontwikkeling van bedrijfsapplicaties.



Componenten binnen het gelaagde architectuurpatroon zijn georganiseerd in horizontale lagen, waarbij elke laag een specifieke rol vervult binnen de applicatie (bijvoorbeeld presentatielogica of bedrijfslogica). Hoewel het gelaagde architectuurpatroon niet het aantal en de typen lagen specificeert die in het patroon moeten voorkomen, bestaan de meeste gelaagde architecturen uit vier standaardlagen: presentatie, business, persistentie en database.

De totale inspanning voor het ontwikkelen van TMSS is ongeveer als volgt verdeeld:
- 35% presentation layer;
- 25% business layer;
- 10% LOFAR internal applications (aanpassen voor de nieuwe interfaces);
- 20% persistence layer;
- 10% database layer.

Elke laag van het gelaagde architectuurpatroon heeft een specifieke rol en verantwoordelijkheid binnen de applicatie. Een presentatielaag zou bijvoorbeeld verantwoordelijk zijn voor het afhandelen van alle gebruikersinterface- en browsercommunicatielogica, terwijl een bedrijfslaag verantwoordelijk zou zijn voor het uitvoeren van specifieke bedrijfsregels die aan het verzoek zijn gekoppeld. Elke laag in de architectuur vormt een abstractie rond het werk dat moet worden gedaan om aan een bepaalde zakelijke vraag te voldoen.

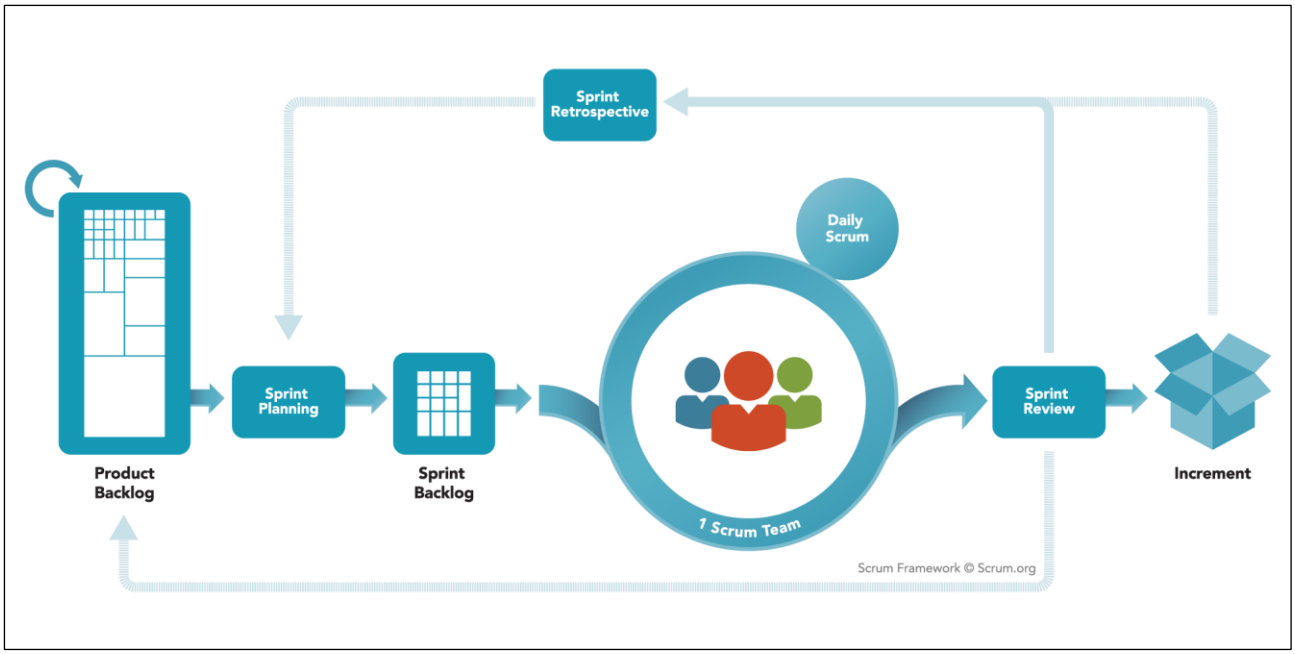
Een van de krachtige kenmerken van het gelaagde architectuurpatroon is de scheiding van verantwoordelijkheden tussen componenten. Componenten binnen een specifieke laag hebben alleen te maken met logica die betrekking heeft op die laag. Componenten in de presentatielaag hebben bijvoorbeeld alleen te maken met presentatielogica, terwijl componenten in de bedrijfslaag alleen met bedrijfslogica te maken hebben.

Een moderne manier om deze gelaagde structuur voor browsergebaseerde applicaties in te vullen is de Model-View-Controller (MVC) architectuur. Dit is een architectonisch patroon dat een toepassing opdeelt in drie logische hoofdcomponenten: het model, de weergave en de controller. Elk van deze componenten is gebouwd om specifieke ontwikkelingsaspecten van een applicatie aan te kunnen. MVC is een van de meest gebruikte industriestandaard webontwikkelingsframeworks om schaalbare en uitbreidbare projecten te creëren.

1.4.4 Software development

Agile SCRUM
De RO ontwikkelteams gebruiken de Agile SCRUM methode. Dit betekent dat dit zelfsturende team in drie wekelijkse sprints werkt.

In de sprint worden planningsactiviteiten (“stories”) voor de sprint overgenomen uit de projectbacklog. De product owner bepaalt waaraan gewerkt wordt door de prioriteit van de backlog items te bepalen. Tijdens de sprint worden de backlog verfijningen gehouden om de backlog klaar te maken voor de volgende sprint. De sprint wordt afgesloten met een sprintreview en een retrospective. Elke sprint eindigt met een werkbaar product dat wordt gedemonstreerd in de sprintreview.

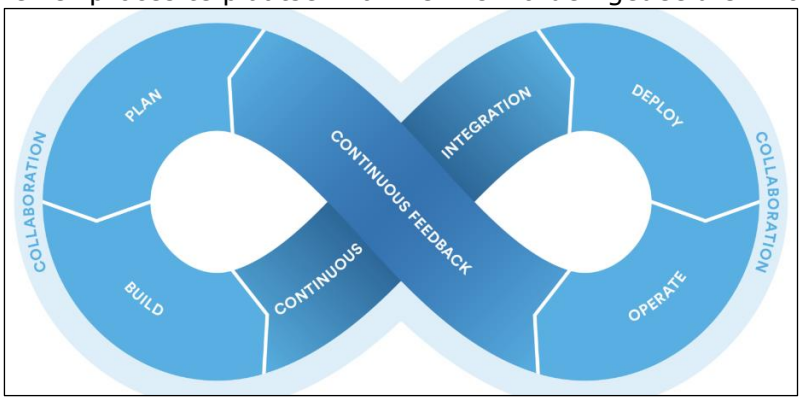


De product owner bepaalt welke items uit de TMSS backlog gepland worden in de volgende sprint van het ontwikkelteam van TMSS.
DevOps
Door moderne technologieën toe te passen, zoals DevOps bepleit, wordt er meer tijd besteed aan softwareontwikkeling en worden de onderliggende systemen efficiënter gebruikt en onderhouden, wat op termijn tot lagere kosten zou leiden. Het huidige softwareontwikkelingsproces bij de RO maakt het moeilijk om veranderingen veilig, snel en

duurzaam in productie te brengen. Dit geldt zowel voor softwarewijzigingen als voor de onderliggende systemen. Het testen van eventuele wijzigingen duurt lang. En het implementeren van wijzigingen zal samen met SOS en de waarnemers moeten worden gepland, omdat de implementatie en verificatie lang duurt.

Door de meeste fasen te automatiseren, werkt de hele pijplijn sneller en met minder fouten. Elke wijziging doorloopt meerdere fasen die kunnen worden gezien als een pijplijn, en elke fase kan de implementatie van de wijziging vertragen. Wanneer code wordt ingecheckt, moet deze worden gebouwd en getest. Het moet worden verpakt en vrijgegeven met de juiste configuratie. Als alleen de toegevoegde wijziging wordt getest, kunnen nieuw geïntroduceerde bugs worden gemist. Om goed te kunnen testen, moeten alle functies in de software worden getest. En in een groeiend systeem kost dit na verloop van tijd meer werk.

Om elke fase goed te doen is zowel expertise van Development als Operations nodig, dus samenwerking tussen beide groepen is nodig. Wanneer een OS-upgrade nodig is, moet alle software opnieuw worden getest. Wanneer een nieuwe systeembibliotheek nodig is, moet deze worden gecontroleerd of deze compatibel is met andere bibliotheken. Wanneer een configuratie op een webserver moet worden gewijzigd voor een nieuwe functie, moet worden gecontroleerd of dit beveiligingsproblemen veroorzaakt. Dus een verandering aan de ene kant kan ook veranderingen aan de andere kant veroorzaken. Als dit niet synchroon loopt, kan dit voor veel vertragingen zorgen. Door mensen uit beide groepen in het Scrum-team en het reviewproces te plaatsen kan kennis worden gedeeld en worden vertragingen geminimaliseerd.



Life Cycle Management kan helpen om het grotere geheel bij te houden om onnodige vertragingen te voorkomen. Software staat meestal niet op zichzelf en moet lange tijd draaien, en door niet naar het grotere geheel te kijken, kunnen veel verrassingen vertragingen veroorzaken bij implementaties. Software zal waarschijnlijk veranderen, zowel op applicatieniveau als op systeemniveau. Hardware heeft ook een beperkte levensduur en moet mogelijk worden vervangen. Het wordt belangrijk om deze potentiële veranderingen te kennen, maar ze ook te volgen.

Binnen de RO zijn veel van de fasen al geautomatiseerd, maar andere fasen moeten nog worden geautomatiseerd.

Zowel Ops als Dev zouden dagelijks moeten samenwerken. Ze moeten allebei deel uitmaken van het Scrum-team. Tijdens de ontwikkeling van features zijn beide expertises nodig. Om ervoor te zorgen dat beide betrokken zijn, moeten beoordelingen worden gedaan door leden van beide expertises. Dit moet worden opgenomen in de workflow van het team.

Door beide groepen in hetzelfde Scrum-team te hebben, zullen beide deelnemen aan alle Scrum-evenementen en vroege feedbackmomenten faciliteren. Tijdens de Sprint Planning

kunnen zij vanuit hun eigen expertise input geven. Ook zullen tijdens de Daily Standups de voortgang en problemen worden besproken om Sprint-items als een team te kunnen voltooien. Tijdens de review wordt de teamprestatie gedemonstreerd en kunnen ideeën voor de volgende Sprint besproken worden. En tot slot is er tijdens de Retrospective tijd om na te denken over wat goed ging en wat beter kan. De goed gedefinieerde structuur van Scrum-evenementen zal de communicatie efficiënt verbeteren en problemen vroegtijdig corrigeren.

Om gemakkelijke communicatie en samenwerking te vergemakkelijken, moet het aantal gebruikte technologieën worden beperkt. Op deze manier wordt de overlap gemaximaliseerd en de leerinvestering voor beide groepen geminimaliseerd. Dit kan door de technologieën te beperken tot Ansible-playbooks, Docker-bestanden en Kubernetes-configuratiebestanden.

1.5 Servicevereisten

De tenderer heeft bewezen dat:

- Hij voldoende ervaring heeft op het gebied van software engineering en informatica voor het leveren van de gewenste producten. *(50 punten)*
- Aantoonbaar werkervaring heeft in twee moderne ontwikkeltalen, met een voorkeur voor Python en Javascript. *(80 punten)*
- Werkervaring heeft in het gebruik van web technologieën en Javascript frameworks, met een voorkeur voor React en Jest. *(80 punten)*
- Werkervaring met agile/SCRUM. *(60 punten)*
- Werkervaring in een wetenschappelijke organisatie, met een voorkeur voor organisaties werkzaam in radio-astronomie. *(80 punten)*
- Kennis van processen rondom gecontroleerde software ontwikkeling. *(50 punten)*

De tenderer erkent dat hij in staat is:

- Te werken in een multinationaal en multicultureel team, geleidt door ASTRON. Dit kan betekenen dat sommige services worden uitgevoerd bij ASTRON.